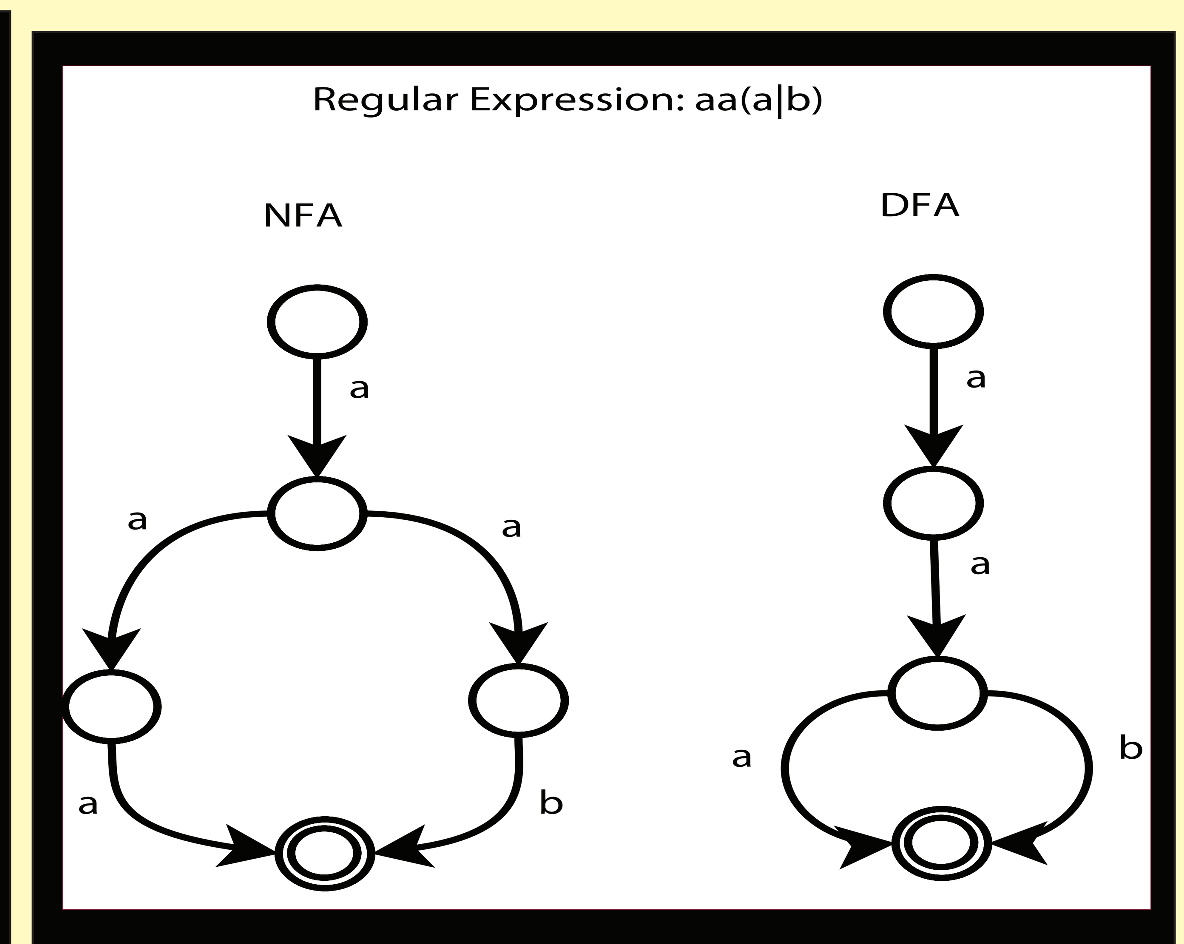
GPGPU Acceleration of Regular Expressions

Regular Expression Overview

Define a set of strings using a compact description.
- Eg. the exact word ‘cat’: cat; all strings containing the word cat: .*cat.*; the characters ‘cat’ in that order anywhere in a string: .*c.*a.*t.*
- Applications include document search, syntax highlighting, network security, and many others. Languages and tools include grep, vi, Perl, Java, awk, and many others.
- Most implementations add convenience operators, or extend the expressive power.

Formally, given a finite alphabet, the following are defined:
- Constants: empty set, empty string, literal characters in the alphabet
- Operations: concatenation, alternation, Kleene star (0 or more repetitions)

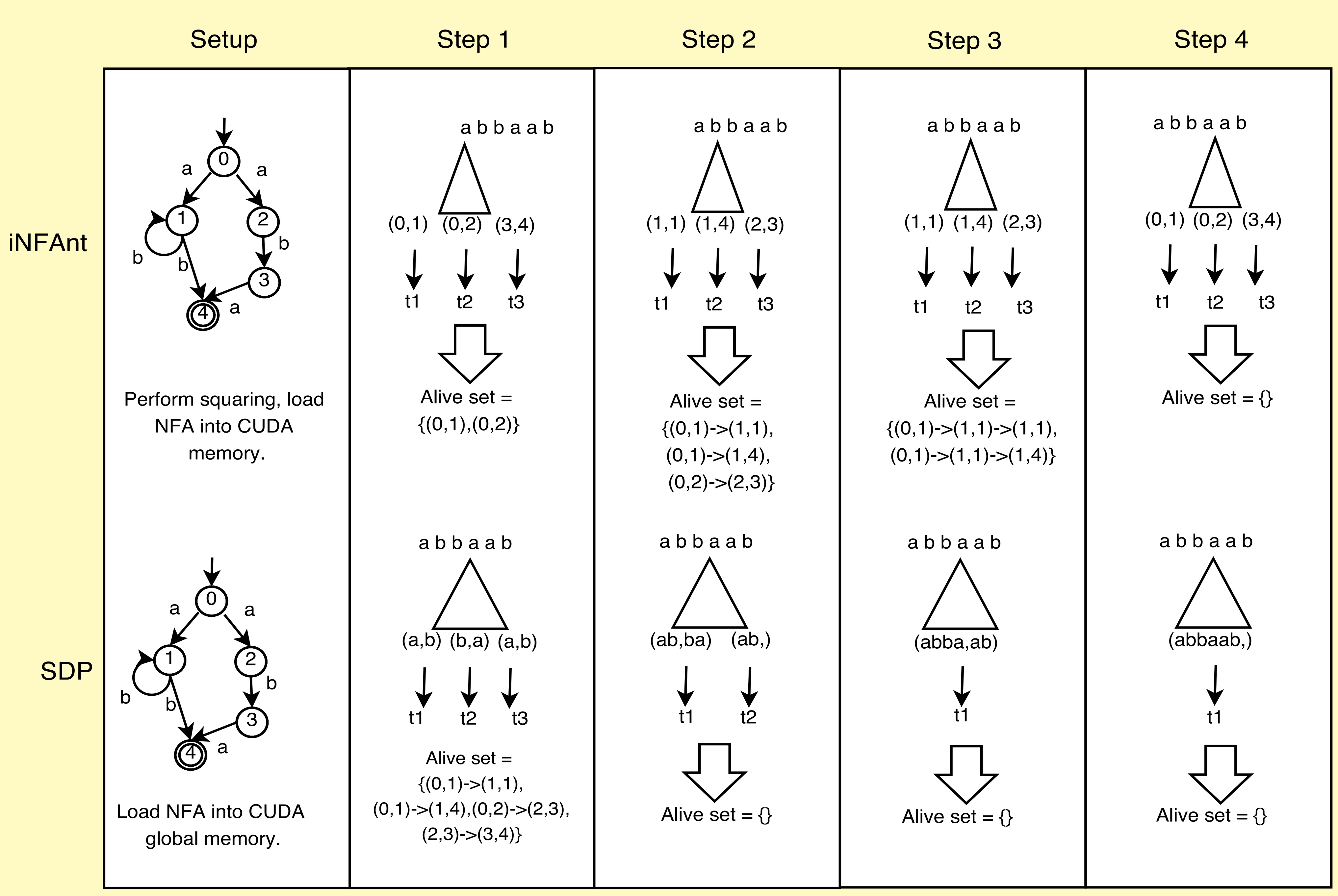
Two equivalent representations:
- Non-deterministic Finite Automata (NFA): A label may appear on multiple outgoing edges.
- Deterministic Finite Automata (DFA): A label may appear on at most one outgoing edge.



Regular Expression Evaluation Background

Three basic evaluation approaches:
- Explicit DFA: Incurs construction cost, but much fastest to process (O(n), where n is the size of the input string).
- Implicit DFA: no construction cost, O(nm) run time where m is NFA size.
- NFA w/ backtracking: More expressive, exponential run time.

	Setup	t <	t ≥
Explicit DFA	$O(2^m)$	$O(n)$	$O(n)$
Implicit DFA	-	$O(nm)$	$O(nm)$
Backtracking NFA	-	$O(2^m)$	$O(2^m)$
iNFAnt	Squaring Time (CPU)	$O(\frac{b}{t} * n)$	$O(n)$
SDP	-	$O(b * \frac{n}{t} * log(n))$	$O(b * log(n))$
Combined	Time To Choose	$O(\frac{b}{t} * n)$	$O(min(n, b * log(n)))$



Future Work

- Compare iNFAnt and SDP approaches using both CPU and GPU threads.
- Develop combined implementation which switches based on NFA complexity and input string size.
- Compare combined approach with standalone and unaccelerated implementations.
- Explore integration with existing regular expression engines.

Parallel NFA Processing: Two Approaches

iNFAnt - Parallel processing of edges.

- General algorithm:
    - 0) “Square” regular expression NFA, if desired (CPU)
    - 1) For each character: (CPU)
        - 1a) Get the list of NFA edges labelled with that character
        - 1b) Spawn a CUDA thread to check each edge for “liveness”.
    - 2) CUDA Kernel: Edge is live if it connects to existing live edges.
    - 3) Match if live edges exist at end of string.
- A thread processes a single transition for the current character.
- Very simple Cuda kernels.
- O((b/t) * n) processing time in general.
- When # of threads t >= branching factor b, O(n) performance.

SDP Approach - Parallel processing of characters.

- General algorithm:
    - Until entire string is processed, do:
        - 1) Break string up into paired substrings (CPU)
        - 2) CUDA Kernel: Check all edges between a substring pair to see if “live” path-segments connect.
        - 3) Match if a live path-segment exists after entire string is processed.
- A thread processes all transitions between pairs of substrings.
- More complex kernels
- O(b * n/t * log(n)) Processing time in general.
- When # of threads t >= n, O(b * log n) processing time.

Combined Approach

Determine the best acceleration approach on the fly:
- When b > n / log(n), use iNFAnt
- When b < n / log(n), use SDP

Contacts:

Chris Strasburg,	cstras@ameslab.gov
Swaroop Dhulpet,	swaroop@cs.iastate.edu
Samik Basu,	sbasu@cs.iastate.edu
Johnny Wong,	wong@cs.iastate.edu
Mark Gordon,	mark@si.msg.chem.iastate.edu